# Ansibleについて知ろう

第1.0版　2024年05月24日

# 目次

# 1.目的

# 目的



- 業務で使うツールなので、知識を身に付ける

- IT自動化に関する知識やスキルに興味があるので、
  業務で触れたAnsibleを通して、その一端を身に付ける

- 勉強の過程で知っておくべきことを共有する

# 2.Ansibleとは

# IT自動化ツールとは



・例えば、「アプリケーションをリリースする」作業の場合、
以下の作業が考えられる
⇒必要なファイルの配置、サービスの再起動、ツールの設定更新など

・挙げた作業を全て手動で実施しようとすると以下のようなことが考えられる
⇒決まり切った手順書の微調整、スケジュールと作業者の日程調整、
　サーバーの状況を常に配慮しながらの長時間の気の張る作業など
⇒**手動での作業はとても面倒**

・決まっているが複雑で面倒な構成作業を自動化してくれるのが**IT自動化ツール**
⇒Ansibleはその一つ

# Ansibleとは



・IT自動化ツールの中でも、単純かつ使いやすいことを目標としたツール

・開発者、システム管理者、ITマネージャーなどのあらゆる分野のユーザーが
使いやすいように設計されている

・一般的にはサーバーが対象だが、仮想環境・ネットワーク機器などにも対応可能

# 3.Ansibleの基本

# Ansibleの概要



- 動作環境：実行側には、Python 2 (バージョン 2.7) または Python 3 (バージョン 3.5 以上) がインストールされていればどのマシンからでも実行可能(ただし、Windowsはサポートがないので注意)。また、処理対象側にもPython 2 (バージョン 2.6以降) または Python 3 (バージョン 3.5 以降)が必要

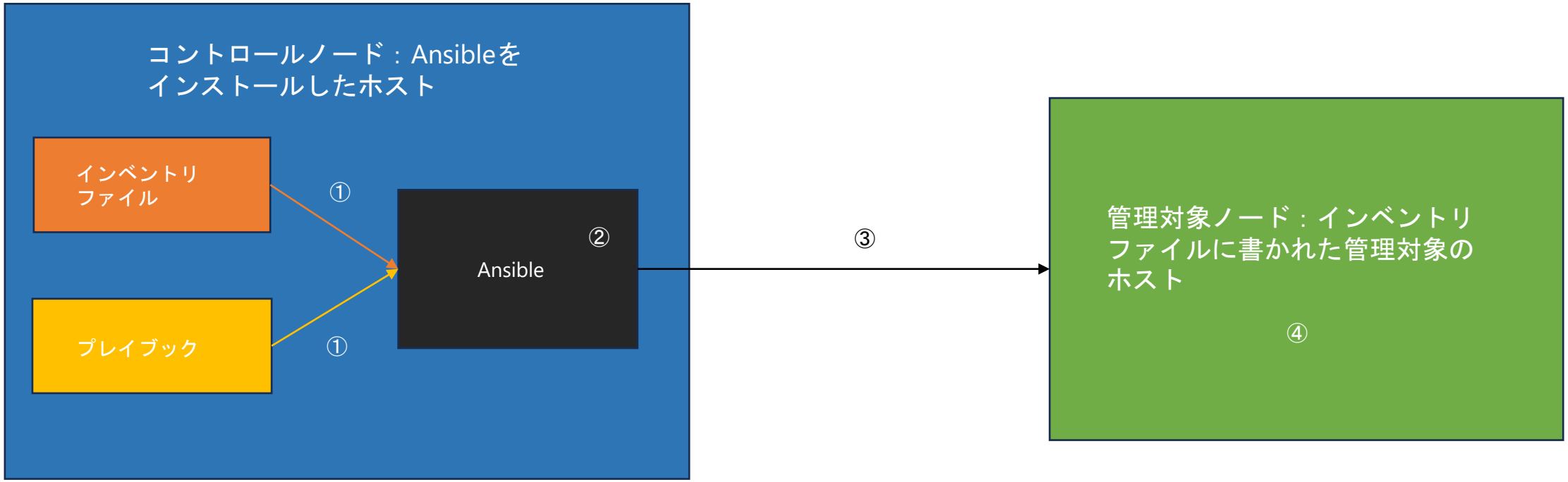
- 定義ファイル書式：YAMLで記述、インベントリファイル(機器一覧表)に関してはini形式で記述可能

- 処理対象へのエージェントのインストール：不要、実行側にAnsibleをインストールすれば使用可能

- 冪等性：処理を何回実行しても実行結果が変わらない、Ansibleは実行時にその処理が実行済みかどうか確認して未実行であれば実行する

# Ansibleの動作






① イベントリファイルとプレイブックをAnsibleに読み込ませる
② Ansibleはプレイブックを Pythonに変換する
③ Ansibleはインベントリファイルに記載された管理対象ノードに、②で作成したPythonのコードをsftpまたはscpで送信する
④ インベントリファイルで記載された管理対象ノードは受信したPythonのコードを実行する

**処理はAnsible側が引き受けてくれるので、ユーザーは実行したい処理に合わせて設定ファイルを用意するだけ**

# インベントリファイル



- Ansibleの処理対象のホストを記述する「機器一覧表」
- ホスト同士の関係を元に記述する
- 実行コマンドで指定するか、ansible.cfgに設定することで、Ansibleに読み込ませることができる

# プレイブック



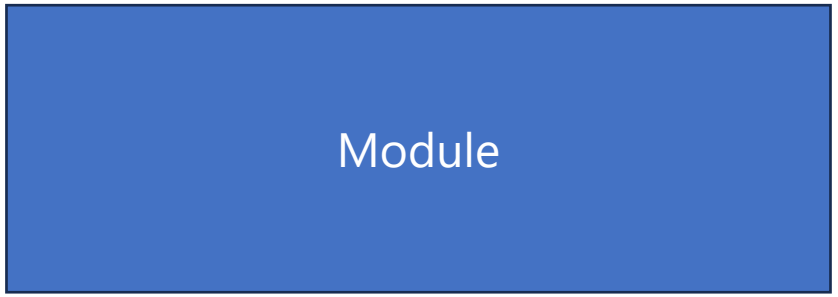
- Ansibleの処理手順を記述し、Ansibleの処理のメインになる「手順書」

- 処理は上から実行されるので、順番を考えてロジックを設計する必要がある

# モジュール



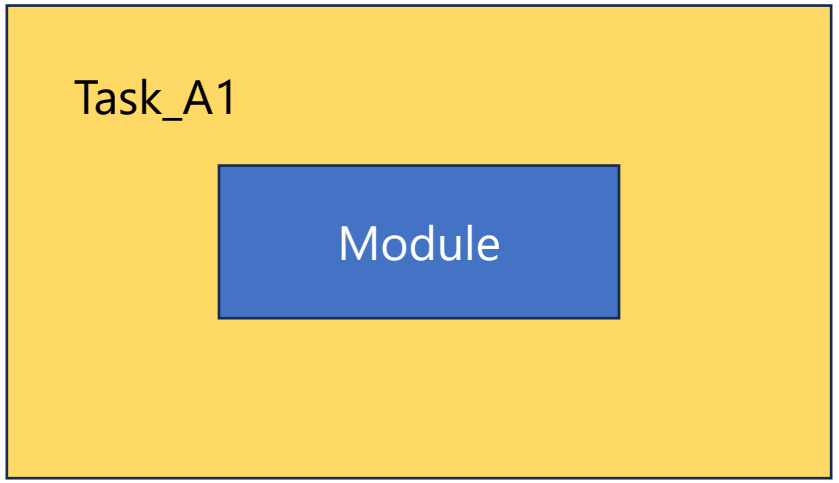
- プレイブックを構成する最小単位で、引数に値を渡さないと機能しない
- 管理対象ノードでリソース制御やシステムコマンドが実行され、結果が戻り値として取得される
- **Ansibleには多くのモジュールが用意されており、それから実行したい処理を選んでいくのが基本(Index of all Modules)**



# タスク



- モジュールに引数を設定して実行可能にしたもの
- 固有の「タスク名」を設定する
- タスクを組み合わせて処理を作成する



# プレイ



- タスクを処理の順番に並べ、対象の管理対象ノード、変数などを追加したもの

- 「プレイ名」を設定する

# プレイ内構造



```yaml
- name: Install Apache. ①
  hosts: takeyamachi
  become: yes
  gather_facts: no

  vars:    ②
    apache_package_name: httpd
    apache_service_name: httpd.service

  tasks:   ③
    - name: Install the httpd package.
      ansible.builtin.yum:
        name: "{{ apache_package_name }}"
        state: present
      notify:
        - start apache service

    - name: Install the firewalld package.
      ansible.builtin.yum:
        name: firewalld
        state: present

    - name: Start "firewalld.service".
      ansible.builtin.systemd:
        name: firewalld.service
        enabled: yes
        state: started

  handlers:   ④
    - name: Change the listening port.
      ansible.builtin.lineinfile:
        path: /etc/httpd/conf/httpd.conf
        regexp: "^Listen "
        line: "Listen 8080"
        validate: httpd -t -f %s
      listen:
        - start apache service

    - name: Start "httpd.service".
      ansible.builtin.systemd:
        name: "{{ apache_service_name }}"
        enabled: yes
        state: started
      listen:
        - start apache service

    - name: Drill a hole for the http port.
      ansible.posix.firewalld:
        port: "{{ http_port }}/tcp"
        permanent: yes
        immediate: yes
        state: enabled
      listen:
        - start apache service
```

① targetセクション：処理の対象になる管理対象ノードや設定を指定する

② varsセクション：処理内で使用する変数を設定する

③ tasksセクション：管理対象ノードで実行する処理を上から順番に記述する

④ handlersセクション：tasksセクション内のタスクが終了した後、taskセクションの実行状況により、実行・未実行になる

# 実行結果



```
y_mrok@ctrl:~/code/exam4$ ansible-playbook -i hosts.yml install_apache.yml

PLAY [Install Apache.] *********************************************************

TASK [Install the httpd package.] **********************************************
changed: [takeyamachi]

TASK [Install the firewalld package.] ******************************************
ok: [takeyamachi]

TASK [Start "firewalld.service".] **********************************************
changed: [takeyamachi]

RUNNING HANDLER [Change the listening port.] ***********************************
changed: [takeyamachi]

RUNNING HANDLER [Start "httpd.service".] ***************************************
changed: [takeyamachi]

RUNNING HANDLER [Drill a hole for the http port.] ******************************
changed: [takeyamachi]

PLAY RECAP *********************************************************************
takeyamachi                : ok=6    changed=5    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0

y_mrok@ctrl:~/code/exam4$
```

# 参考サイト



- Ansibleの使い方
- Ansibleとは？概要から活用例、インストール方法までわかりやすく解説
- Ansible ドキュメント

# EOF